VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Implementace hry - Magic

Implementation Magic Game

2013 Lukáš Holiš

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Lukáš Holiš**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Implementace hry - Magic
Implementation Magic Game

Zásady pro vypracování:

Cílem této bakalářské práce je zaměřit se na vývoj herních aplikací pro mobilní telefony Android, či osobní počítače, pro více hráčů spojených přes herní server. Výsledkem bude implementace serverové časti hry a protokolu pro podporu multiplayer hry a otestování navrženého protokolu na ukázkové hře na osobním počítači.
1. Návrh a implementace serverové části hry a komunikačního protokolu tak, aby podporovala hru z osobního počítače i z telefonu se systémem Android. Sofistikované zabezpečení serveru a protokolu, ať se předejde podvrhům. Zabezpečení způsobu hry ze strany serveru a protokolu na výpadky sítě, odhlášení hráče z rozehrané hry, a podobně.
2. Návrh a implementace hry Magic na osobním počítači s využitím serveru.
3. Otestování dané hry a serveru a zhodnocení použitého postupu vývoje.
Hra Magic je tahová hra. V rámci této hry stojí proti sobě hráči, z nichž každý vlastní svou vesnici a hrad, kdy pomocí karet buduje vesnici a armádu. Hráči poté proti sobě stojí na bojišti a prohrává ten, jehož hrad padne jako první.

Tato bakalářská práce se zaměřuje na serverovou část hry, komunikační protokol a herního klienta pro osobní počítač. Využívá umělou inteligenci protivníka z práce druhého studenta.

Seznam doporučené odborné literatury:

BURNETTE, Ed. Hello, android: introducing Goggle's mobile development platform. 3rd ed. Raleigh, N.C.: Pragmatic Programmers, 2010, xviii, 293 p. ISBN 978-193-4356-562.
DEITEL, Paul J. Android for programmers: an App-driven approach. 3rd ed. London: Pearson Education [distributor], c2012, xxx, 781 p. ISBN 01-321-2136-0.
LEE, Wei-Meng H. Beginning android 4 application development: an App-driven approach. 1st ed. Indianapolis, IN: Wiley Pub., Inc., 2012, p. cm. ISBN 11-181-9954-5.
MEIER, Reto. Professional Android 4 application development: an App-driven approach. Updated for Android 4. Indianapolis: John Wiley, 2012, xlii, 817 p. ISBN 978-111-8262-153.
SATYA KOMATINENI, Dave MacLean a Eric Franchomme TECHNICAL REVIEWERS. Pro Android 4: an App-driven approach. Updated for Android 4. New York: Apress, 2012, xlii, 817 p. ISBN 978-143-0239-307

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jakub Štolfa**

Datum zadání: 16.11.2012

Datum odevzdání: 07.05.2013

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 30. dubna 2013

Lukáš Holík

.....

Tímto bych rád poděkoval všem, kteří mi s prací pomohli, jmenovitě vedoucímu bakalářské práce Jakubu Štolfovi za profesionální rady a organizátorské schopnosti, Michalu Seidlerovi a Michalu Holiši za technické rady. Bez nich by práce nebyla nikdy dokončena.

Abstrakt

Tato bakalářská práce se skládá ze tří částí. První část bakalářské práce se zabývá návrhem a implementací serverové a klientské části hry a vzájemné komunikace mezi nimi. Zabezpečením komunikace od případných útoků nebo nečekaných událostí. Vše je finálně navrženo tak, aby byla možnost spustit aplikaci jak na mobilním telefonu se systémem Android, tak na osobním počítači. Hra obsahuje hru více hráčů a je uzpůsobena tak, aby byla zajištěna komunikace mezi všemi účastníky. Serverová část aplikace je společně s databází nasazena na speciálně upraveném virtuálním serveru. Ten slouží jako databázový a herní server zároveň.

Druhá část bakalářské práce, se zaobírá implementací samotné hry na osobním počítači společně s umělou inteligencí. Umělá inteligence má být převzata od druhého studenta Filipa Krupíka, který se podílí na tvorbě projektu.

Třetí část bakalářské práce, spočívá v otestování navržených součástí a zhodnocení použitého postupu a vývoje.

Klíčová slova: klient-server architektura, PC hra, zabezpečení komunikace TCP, SSL

Abstract

This thesis consists of three parts. The first part of the thesis deals with the design and implementation of server and client sides of the game and the interaction between them. Communication security from possible attacks or unexpected events. Everything is finally designed with the ability to run as an application on a mobile phone running Android, as well as on the personal computer. The game contains a multiplayer game it is designed as to ensure communication between all participants. The server portion of the application, together with a database deployed on a special virtual server. This server serves as database and game server together.

The second part of the thesis is concerned with the implementation of the game itself on a personal computer with artificial intelligence. Artificial intelligence has to be taken from another student Filip Krupík, which is involved in creating the project.

The third part of the thesis consists in part of the proposed test and evaluation procedure used in development.

Keywords: client-server architecture, PC game, TCP communication security, SSL

Seznam použitých zkratek a symbolů

API	–	V programováni se jedná o jednu nebo více knihoven, které poskytují nové funkce a možnosti.
SSL	–	Secure sockets layer - jedná se o vrstvu vloženou mezi transportní vrstvu (v mém případě TCP/IP) a aplikační vrstvu pro zabezpečení komunikace.
TBCG	–	Turn based Card game - tahová karetní hra
ping	–	Jedná se o odezvu, čas jak dlouho trvá přenést pakety z klienta na server a opačně.
Multiplayer	–	Hra více hráčů.
output stream	–	Data zasílaná do streamu(proudu)
input stream	–	Data čtená ze streamu(proudu)
OS	–	Operační systém

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Jelikož se v poslední době stále více zdokonalují jak mobilní telefony, tak osobní počítače, stoupají tudíž i nároky uživatelů na aplikace vyvíjené na tyto zařízení. Proto jsem se rozhodl společně se svým kolegou Filipem Krupíkem, vyvinout hru, která bude spustitelná jak na počítači, tak na mobilním telefonu s operačním systémem Android.

Mluvíme-li o mobilních telefonech, většina z nich má dnes dotykovou obrazovku, což je pro náš vývoj ideální z důvodu velké škály možností, které dotyková obrazovka poskytuje. Víme, že velká část obyvatelstva vlastnící osobní počítač nebo moderní mobilní telefon s dotykovým displejem, jej používá ke hraní her. Proto při výběru žánru hry byl určen takový, který by mohl být realizovatelný jak na mobilním telefonu, tak na počítači. Jedná se o karetní hru s fantazy námětem, která má oficiální název Magic. Pracovní a používaný název, který jsme společně odsouhlasili na Clash of the Lords (CotL). Hra obsahuje speciální prvky, kterými se chceme odlišit od ostatních podobných her. Podrobným popisem hry se budu zabývat v kapitole 2.1.2 Žánr a provedení samotné hry.

Dalším požadavkem kladeným uživateli je, aby hra pokud možno byla spustitelná jak na osobním počítači, tak na mobilním telefonu. Pro vývoj jsme se tudíž rozhodli použít programovací jazyk Java pro serverovou část a c# pro klientskou část, jelikož se jedná o silné programovací jazyky vhodné pro náš projekt.

Mezi další nároky kladené na dnešní moderní aplikace, je možnost hrání hry online s ostatními hráči. Proto při vytváření hry byla brána zřetel na to, aby se hráčům dostavilo vzájemné rivality a hra tudíž obsahovala Multiplayer (hra více hráčů), více v sekci 2.1.3 Hra více hráčů. K tomu bylo zapotřebí, aby hra používala klient - server architekturu a byla tudíž řízena centrálním serverem. Tento server pak zajišťuje synchronizaci a zároveň šifrovanou komunikaci pomocí SSL, všech napojených klientů hrajících společně v předem vytvořené místnosti. Dále je server naimplementován tak, aby dokázal zasílat požadavky nebo odpovídat na požadavky pomocí komunikačních balíčků. Tyto balíčky jsou zašifrované pomocí asynchronní šifry, tudíž jsou zasílaná data chráněna a bezpečně zasílána z jednoho zařízení na druhé. Další částí, kterou se má bakalářská práce zabývá je vývoj desktopové verze hry. Zde jsem využil základní poznatky mého kolegy, který pracuje na Android aplikaci.

2 Idea hry Magic (CotL)

Jak si můžeme všimnout, dotykové telefony v poslední době zcela nahrazují obyčejné telefony s integrovanými pevnými tlačítky, které na trhu pomalu ale jistě zcela vymizí. Není divu, že se tomu tak děje. Tato technologie je mnohem prozíravější například k vůli tomu, že ne vždy potřebujeme mít k dispozici klávesnici, když si zrovna čteme email nebo sledujeme video. Jelikož se vývoj elektroniky stále neúprosným tempem žene dopředu a zařízení, které kupujeme se stávají menšími a mnohem výkonnějšími, kladou se tímto větší nároky i na vývoj aplikací.

2.1 Návrh hry

Před tím než jsme se společně se svým kolegou rozhodli vytvořit hru, stanovili jsme si požadavky, které na hru klademe a podle toho jsme dále postupovali.

1. Jaký operační systém zvolit?
2. Jaký žánr a provedení hry se ujme?
3. Hra více hráčů?
4. Programovací nástroje a jazyky.

2.1.1 Operační systém telefonu

Zde jsme se zamysleli nad tím, na který operační systém telefonu, dále jen OS, vyvíjet. Proto jsme si nejprve zjistili rozprostření mobilních OS na trhu, kde jsme zjistili podle tabulky č. 1 agentury IDC, že v roce 2011 byl nejoblíbenějším, či nejvíce lidmi vlastněným mobilním OS Android. Nadále v budoucnu má být stále nejvíce zastoupeným mobilním OS na trhu. Windows Phone má být podle průzkumů druhým nejpoužívanějším OS v roce 2015.

Jelikož byl Android vyvíjen z jádra Linuxu, jedná se o open source OS, který může být spuštěn na jakémkoliv zařízení bez ohledu na chipset procesoru a velikosti rozlišení. Navíc se jedná většinou o levná zařízení a tudíž pro uživatele cenově lákavější, než na příklad iOS (Apple) produkty, které se pohybují většinou v dražší cenové kategorii. Proto Android převládá jak v mobilních telefonech tak i tabletech, které se poslední dobou stávají čím dál víc atraktivnějšími. Proto jsme se rozhodli vyvíjet na Android platformu.

2.1.2 Žánr a provedení samotné hry

Při volbě žánru hry jsme čerpali z vlastních zkušeností, podle toho co nás baví a chtěli jsme to okořenit něčím co v dnešním herním průmyslu chybí. Již od počátku jsme chtěli hru realizovat na základě karet a inspiraci jsme hledali u slavné karetní hry, která je jak stolní tak počítačovou hrou Magic the Gathering. Jedná se o styl hry Card Game - karetní hra. Zde proti sobě stojí vždy dva hráči se svými balíčky karet, které obsahují karty zdrojů surovin, potřebné k následnému používání ostatních karet, které danou surovinu

Operační systém	Podíl na trhu 2011	Podíl na trhu 2015	2011-2015 změna
Google Android	39,5%	45.4%	23.8%
BlackBerry OS	14.9%	13.7%	17.1%
iOS	15.7%	15.3%	18.8%
Symbian	20.9%	0.2%	-65.0%
Windows Phone 7(+WM)	5.5%	20.6%	67.1%
Ostatní	3.5%	4.6%	28.0%

Tabulka 1: Podíl mobilních operačních systémů na trhu v blízké budoucnosti.

vyžadují.

Chtěli jsme naši hru trochu obohatit o prvky, které hra Magic the Gathering postrádá a vytvořit tak nezávislý herní žánr tahová karetní hra s prvky strategie(Turn Based Strategy Card Game). Co se týče karet, je nutno říci, že je dělíme na dva druhy a to stavitelské a bojové. Stavitelské karty mají větší bonus k opravě rozbořených budov a zaručují delší setrvání ve hře. Naopak bojové karty mají vetší útočnou sílu se kterýma jednoduše zničíte protivníka.

Hráč po startu hry dostane do ruky pět náhodných karet, tyto karty obsahují jak bonusy, tak útočné číslo. Je tedy otázkou náhody a štěstí, jak mocné karty dostanete, a jak je hodláte použít.

Výsledkem naší práce, nemá být pouze úspěšné zvládnutí bakalářské práce ale také vytvoření herního žánru, který by mohl přinést revoluci hernímu průmyslu, který v současné době upadá a hry se stávají čím dál jednoduššími. Tuto skutečnot se vývojáři snaží zamaskovat perfektně vypadajícím vzhledem, který osloví většinu hráčů ale jedná se pouze o chvilkový dojem a takovéto hry se stávají dříve či později nudnými.

2.1.3 Hra více hráčů

Jednou z věcí, kterou jsme již od začátku chtěli v naší hře mít je hra více hráčů. Jelikož se stále více zařízení na světě připojuje k internetu, roste počet hráčů, kteří mají zájem změřit své síly s ostatními hráči z celého světa. Proto většina moderních her obsahuje hru více hráčů.

Existuje více způsobů jak propojit zařízení nebo hrát s více lidmi naráz.

1. Dva a více hráčů na stejném zařízení

   Výhody:

   - Hráči jsou většinou přátelé, tudíž hra probíhá poklidně, nepadají urážky a podobně.
   - Jednoduché na implementaci, není potřeba použití komunikačních protokolů jako například TCP/IP, UDP a tak dále.
   - Stačí pouze jedno zařízení na hraní a navíc není nutnost připojení k internetu.

Nevýhody:

- Nutnost být na jednom místě a předávat si zařízení. (Ne vždy nevýhodou, hráči totiž sedí spolu jako by hráli stolní hry = udržování kontaktu.)
- Předávání telefonu po každém zahraném kole, může být po chvíli nepříjemné, hlavně ve velkém počtu hráčů.

2. Síťová hra

   Výhody:

   - Mohou hrát všichni hráči, kteří jsou připojeni ke stejné síti, na příklad pomocí LAN nebo WLAN.
   - Může hrát více lidí naráz každý na svém zařízení.
   - Možnost určit kdo má k síti přístup a kdo ne.

   Nevýhody:

   - Nutnost implementace serverové části, kde jeden člověk, který založí hru se stává serverem. To znamená větší zátěž na jeho zařízení.
   - Nutnost existence sítě složené z aktivních a pasivních prvků, které zajišťují komunikaci mezi zařízeními. (aktivní - switch, síťová karta, router a pasivní - koaxiální, optický, UTP kabel)

3. Hra přes internet Výhody:

   - Tisíce hráčů připojených a připravených hrát.
   - Možnost hrát i s lidmi z jiných zemí.
   - Větší konkurence hlavně ve hrách, které obsahují žebříček nejlepších hráčů s nejvyšším skóre v dané hře.

   Nevýhody:

   - Nutnost implementace serverové části stejně jako u Síťové hry, avšak z důvodu odlišných rychlostí připojení, a vzdálenosti, které musí pakety urazit je nutná synchronizace mezi klienty (hráči).
   - Proto dnes, při hostování většího množství hráčů je nutné mít kvalitní servery, které zvládnou vysoký nápor stále dotazujících se hráčů.
   - Připojení k internetu může být problémem na některých místech.

   Naše hra obsahuje hru více hráčů přes internet, kde jsem se střetnul s návrhovým vzorem klient - server komunikace kterému věnuji kapitolu 3.1 Klient - Server architektura.

2.1.4 Programovací nástroje a jazyky

1. Vývoj na mobilní telefon se systémem Android:

   Vývoj aplikací na mobilní telefony ze systémem Android většinou probíhá pomocí programovacího jazyku Java obohacený o Android Software Development Kit (SDK). Jedná se o sdružení nástrojů jako jsou debugger, emulátor, knihovny a zdrojové kódy určené právě pro vývoj aplikací na operační systém Android. Jedná se o bezplatnou verzi SDK a právě díky velkého zájmu jej odkoupila společnost Google, která jej nadále vyvíjí.
   Pro vývoj na mobilní telefon používáme vývojové prostředí Eclipse, které podporuje Android SDK společně i s emulátorem, který nám zajistí otestovat hru na odlišných mobilních zařízeních. Eclipse je také bezplatný program, tudíž je pro vývoj ideální.

2. Vývoj na PC platformu: Jelikož je serverová aplikace spuštěná na virtuálním serveru zřízeném přímo pro tuto aplikaci, který obsahuje Linux OS vyvíjel jsem ji v programovacím jazyku Java, samotnou hru pak v jazyku C# . Pro vývoj v Javě jsem využil bezplatného vývojového prostředí NetBeans a pro vývoj v C# jsem použil Visual Studio 2012.

3 Technologie

Dnes se již prakticky všude setkáváme s aplikacemi, které komunikují skrze vzdálený server s databázovým serverem, který poskytuje data a ty se pak zasílají zpět na klienta, kde se zobrazí na displeji. Takováto aplikace používá takzvanou Klient - Server architekturu (Client - Server Architecture). Zde se pokusím osvětlit, jak tato technologie funguje, jaké má klady, zápory a jaké protokoly lze použít pro přenos dat mezi dvěma zařízeními.

3.1 Klient - Server architektura

Jednoduše řečeno se jedná o síťovou architekturu, provedení, kde je úkolem oddělit klienta (aplikaci) od serveru a navázat mezi nimi komunikaci skrze síť. Nutností existence této architektury je implementace alespoň jednoho serveru a zároveň jednoho či více klientů.

Základem tohoto principu, je zasílání požadavků ze strany klienta na server, který je vyhodnotí, spočítá a odešle výsledek zpět na klienta.Ten se podle výsledku dále rozhodne co dělat. Nejlépe si princip fungování klient - server aplikace ukážeme na příkladu.

Příklad 3.1
Uživatel na internetu objevil Free to Play (bez poplatku, zdarma) počítačovou hru Tera Online. Zaujala ho a chtěl by si ji zahrát. Tudíž se musí zaregistrovat.

V tomto případě je klientskou částí jakýsi herní formulář, který musí nový uživatel vyplnit, aby se mu vytvořil unikátní účet potřebný pro vstup do hry. Tudíž po řádném vyplnění údajů, které formulář žádá, se z klientské aplikace odešle požadavek na příklad "/registrace"na vzdálený server. Server poslouchá příchozí požadavky a má v sobě zavedené co má přesně udělat pro každý příkaz, který na server přijde. V případě požadavku "/registrace":

1. Převezme údaje z formuláře, které byly uživatelem vloženy.
2. Připojí se na vzdálený databázový server a uloží je zde.
3. Odešle potvrzovací email na zadanou emailovou adresu z formuláře
4. V případě úspěchu pošle na klienta odpověď "/hotovo"
5. V případě neúspěchu pošle na klienta odpověď "/chyba"

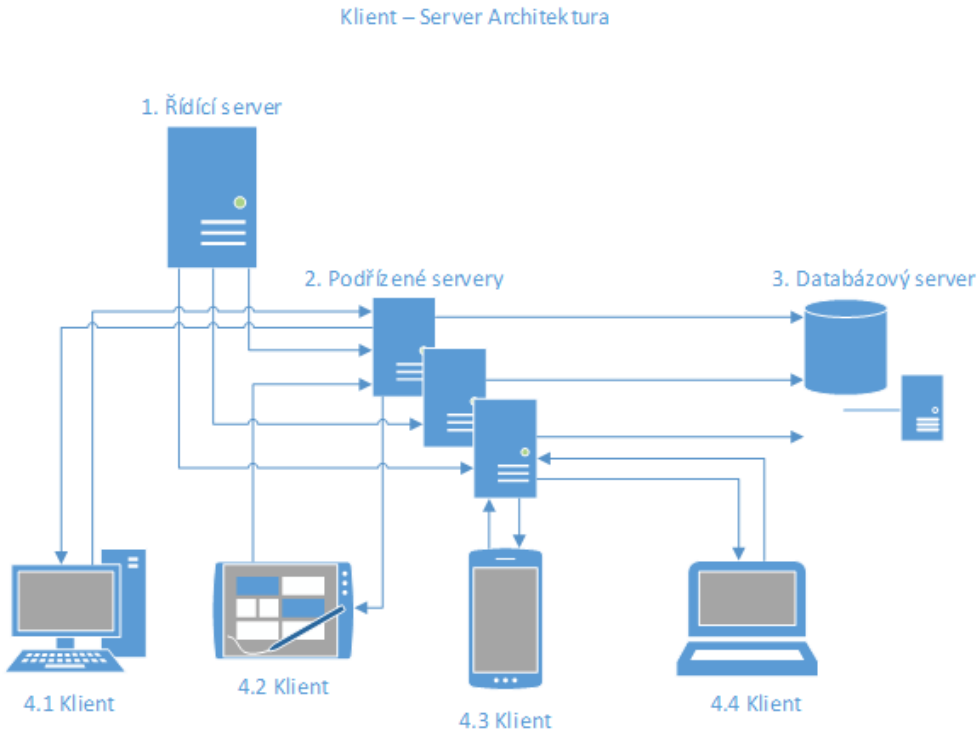
Podle scénáře, který nastane se zobrazí výsledek na klientské straně. V případě číslo 4. zobrazí hlášku "Váš účet byl v pořádku vytvořen, prosím zkontrolujte email, který vám byl zaslán na vaši emailovou adresu". Kdežto v případě číslo 5. můžeme vyvolat jinou událost, na příklad "Zkontrolujte prosím zadaná hesla, obě se musejí shodovat.".

Zde můžeme jednoduše vidět interakci mezi klientem a serverem, kde klient žádá po serveru nějakou akci, server ji vykoná a podle scénáře, který nastane zpětně kontaktuje klienta, který se zařídí podle toho, jakou odpověď dostal.

■

3.1.1 Schéma klient - server architektury

Pro názorné přiblížení, jak může vypadat takové rozprostření komponent klient - server aplikace slouží obrázek č.1.



Obrázek 1: Klient - Server Architektura - obecné schéma komponent

Server všeobecně:
Jednoduše se jedná o počítač, který zajišťuje služby nebo program, který je na něm nainstalován a ten zajišťuje tyto služby. Servery bývají umístěné ve speciálních místnostech s klimatizací, které nazýváme serverovny nebo mohou být umístěny v takzvaném racku, což je speciální skříň, která slouží k ušetření místa, ta může být zajištěna proti výpadku elektrického proudu a podobně.
Aplikačně myšleno je většinou server stavěn tak, aby nezávisle mohl přijímat požadavky více klientů naráz, a proto se často setkáváme s implementací serveru, který obslouží nově připojená zařízení v novém vláknu. To znamená, že program dokáže naráz obsluhovat více zařízení bez toho abychom museli tvořit fronty na serveru a podle nich obsluhovat uživatele jeden po druhém v takovém pořadí, ve kterém se dotazovali.
Servery mezi sebou nebo server s klientem komunikují pomocí takzvaných komunikačních protokolů. Tyto protokoly slouží k vyhledání cílového zařízení podle ip adresy a následném pokusu dopravit zaslané pakety na toto zařízení. Ne vždy je tomu tak a podrobně se komunikačními protokoly zabývám v kapitole č. 3.2 Komunikační

protokoly, kde nastíním vlastnosti jednotlivých komunikačních protokolů, které lze použít v klient - server architektuře.
Podrobný popis všech zařízení použitých v obrázku č.1:

1. Řídící server:

   Mluvíme li o velkých projektech, aplikacích, hrách, které denně hostí sta tisíce až miliony uživatelů, je zde jasné, že všechny tyto uživatele nemůže obsloužit jediný server. Je zde zapotřebí rozložit tento nátlak uživatelů, mezi více serverů, které budou řízeny právě jedním hlavním řídícím serverem. Tento server se stará o synchronizaci všech podřízených serverů a komunikaci mezi nimi. V angličtině jej můžeme znát pod jménem Master server.

2. Podřízené servery:

   Jedná se právě o ty servery, mezi které je nátlak uživatelů rozprostřen. V počítačových hrách se můžeme setkat právě s touto situací, kde při prvním připojení do hry nalezneme seznam dostupných serverů. Každý z těchto serverů má jiné umístění, a proto si raději vybíráme servery, které jsou nejblíže našemu místu pobytu a to k vůli co nejmenšímu pingu.

   Právě všechny podřízené servery mají přístup ke stejné databázi, tudíž ke stejným datům. Proto mohou hostit libovolné uživatele, kteří mají vždy k dispozici aktuální informace, at' už kdokoliv z jiného podřízeného serveru změnil cokoliv.

3. Databázový server:

   Většinou nejdůležitější část, kde se všechny důležité data schovávají je právě databázový server. Tento server je přístupný všem podřízeným serverům a data jsou zde sdílená mezi nimi. Databázový server, může hostit mnoho databázových systémů a mezi ně patří na příklad Firebird, MySQL, Oracle XE nebo Microsoft SQL Server. Tyto systémy slouží pravě k ukládání a správu dat, pomocí programovacího jazyka SQL.

4. Klienti:

   Ze schématu je názorné, že ne každý klient musí nutně přistupovat ze stejného zařízení. Pokud je aplikace vyvíjena na vícero zařízení, klienti mohou bez problému přistupovat ze všech těchto platforem. Ze schématu, klient 4.1 používající osobní počítač může bez problému zasílat požadavky na server ve stejný okamžik, jako zasílá klient 4.2 který je připojený přes tablet. K tomu slouží právě takzvaný multithreading (více vláknová aplikace).

Závěrem k této podkapitole chci dodat, že architektura klient - server je dostatečně sofistický přístup, který je vhodné použít pro vývoj hry. Lépe řečeno multiplayer části, kde je potřeba přenášet data mezi jednotlivými klienty tak, aby vždy každý hráč který hraje hru s jinými hráči, věděl o tom co se právě děje. Proto považuji tuto architekturu za nejdůležitější část při vývoji multiplayer části.

3.2 Komunikační protokoly

Díky komunikačním protokolům můžeme přenášet data mezi dvěma zařízeními (počítači) podle předem stanovených regulí. Jedná se o určené pravidla, syntaxi a synchronizaci probíhající komunikace. Protokoly dělíme na dvě části, hardwarové a softwarové.
Nás zajímají protokoly související s komunikací přes internet, a k tomu existuje mnoho protokolů, které si představíme a vybereme z nich ten nejvhodnější.

1. PoP3, IMAP, SMTP
2. Telnet, SSH
3. FTP
4. HTTP
5. UDP
6. TCP

1. PoP3, IMAP, SMTP:

   Zkráceně řečeno jedná se o emailové protokoly, což znamená, že slouží k zasílaní emailu, čtení emailové schránky a komunikaci mezi emailovými servery. Většina lidí má na počítači nainstalovaný emailový klient, jehož práce je právě stahování z poštovních serverů, pomocí protokolů PoP či IMAP. Samotné emaily se po internetu zasílají pomocí SMTP (Simple mail transfer protocol) protokolu.

2. Telnet, SSH:

   Jedná se o dva dnes nejčastěji používané protokoly v případě vzdáleného řízení komunikace sloužící k připojení ke vzdálenému počítači. Telnet protokol je dnes součástí většiny operačních systémů, na příklad u Windows systému program vzdálená plocha (remote desktop). Telnet má však jeden veliký nedostatek a to jest, že není šifrovaný, tudíž jsou data, která v síti zasíláme náchylná na odposlouchávání.

   Protokol SSH tyto nedostatky upravil a proto se dnes nejčastěji používá na příklad při komunikaci se vzdáleným serverem. Funguje přibližně tak, že vytvoří zabezpečený tunel mezi jedním počítačem a druhým počítačem, přičemž můžeme mezi sebou přenášet data, zasílat příkazy pomocí příkazové řádky to vše bezpečně zašifrované.

3. FTP:

   Jedná se o platformě nezávislý protokol pro přenos souborů,který je dnes součástí veškerých webových prohlížečů.Ty zaručí přenos (download/upload) souborů, avšak zde vzniká nebezpečí při přenosu z tohoto důvodu data nejsou zašifrovaná, posílají se jako běžný text. Dnes se v některých sítích používá pouze protokol HTTP.

4. HTTP: Hypertext transfer protocol, aneb protokol pro přenos hypertextových dokumentů, napsaných programovacím jazykem HTML. Přidáme - li k tomuto protokolu XML části, můžeme tak spouštět i vzdálené webové služby, které jsou dnes dost používaným prvek v programování. Na příklad pomocí SOAP serializace, můžeme používat zmiňované web services (webové služby).

   Příklad 3.2
   Dokážeme vytvořit klient server aplikaci, použitím HTML a SOAP?

   Vytvoříme dva Java projekty, jeden klientskou část a druhý serverovou část.

   Serverová část:

   Použít můžeme například GlassFish server (dále jen GF Server), který je volně ke stažení. Můžeme jej nainstalovat jak na lokální disk, tak na vzdálený server. Tento GF server slouží právě k zajištění komunikace mezi klientem a serverem. Dále serverové aplikaci dáme vědět, kde se GF server nachází a kompilovaný kód pak spouštíme přímo na něm.

   Od této chvíle můžeme vytvářet web metody, které budou volány z klientské části tehdy, kdy jsou potřeba. Na příklad hráč se chce přihlásit, tudíž vytvoříme boolean metodu.Ta přijímá dva parametry jméno a heslo pojmenujeme ji login. Tato metoda vrátí true, když bude přihlášení úspěšné, false když nebude. Máme připravenou metodu k použití a tudíž přistoupíme k tvorbě klientské části.

   Klientská část:

   Založíme-li Java projekt a máme serverovou část připravenou, můžeme ji připojit pomocí web service. Od této chvíle, můžeme přistupovat ke všem web metodám umístěným na serveru. Zpátky k naší web metodě login(jméno, heslo). Například pomocí přihlašovacího formuláře s dvěma text boxy a tlačítkem, můžeme na událost stisku tlačítka spustit metodu, které se zavolá vzdáleně na serveru kde se provede a vrátí výsledek, podle kterého se klientská aplikace rozhodne co se bude dále dít.

   Tento příklad měl ukázat možnosti HTTP protokolu obohaceného SOAP protokolem kde se může zdát, že jsou právě tyto dva protokoly dokonalé pro náš projekt, ovšem opak je pravdou.

   Závěr:

   Nejprve jsem si myslel, že by to bylo v našem případě dostačující. 0všem po delším zkoumání a zjišťování vyšlo najevo, že scénář kdy jeden hráč použije kartu na druhého hráče, kde musíme zajistit komunikaci mezi dvěma hráči, je nerealizovatelný. Jde právě o to, že nemáme přístup ke komunikačnímu kanálu daných klientů a vše je realizováno právě skrze webové služby, které se starají o komunikaci. Jedná se zde tudíž o jednocestnou komunikaci z klienta na server a zpátky. Další nevýhodou je že převod do XML a parsování klade vetší nároky na

procesor a proto je možné, že by se aplikace mohla po nějaké době sekat nebo by mohla být pomalá na reakce hráčů. Pro naše účely je tudíž bohužel nedostačující.

■

5. UDP: Tento protokol je známý tím, že nezaručuje na sto procent, že se zaslaný datagram dostaví do cíle. Je možné, že se na cestě k cíli ztratí nebo data přijdou v jiném pořadí nebo se doručí vícekrát. Tento protokol se hodí na místech kde zasíláme jednoduchá data, binární čísla a tak dále. Po delším zkoumání jsem zjistil, že pomocí tohoto protokolu je psáno velké množství her, především FPS (First person shooter) akční střílečky. Složitost programování UDP protokolu je taktéž na jiné úrovni, neexistují zde pomocné knihovny, tudíž si ji prakticky člověk musí naprogramovat od zakladu podle toho, jak se mu to hodí. Navíc UDP je vhodné použít tam, kde potřebujeme v reálném čase přenášet informace, například pohyb postavy po mapě. V našem projektu není potřeba přenášet data v reálném čase, tudíž menší zpoždění nehraje roli. Posledním protokolem, který můžeme použít a je pro nás nejideálnější je TCP.

6. TCP: Na rozdíl od UDP je TCP protokol stavový což znamená, že data odeslaná na danou adresu a port, vždy dorazí do cíle. Před začátkem komunikace se totiž vytvoří pomyslný tunel mezi oběma účastníky, po kterém proudí data oběma směry. TCP protokol tudíž zaručuje, že se odeslaná data doručí jak ve stejném pořadí, tak vždy ke správnému příjemci.

   Prvky zpomalující komunikaci u TCP spojení:

   (a) Založení komunikačního kanálu tak zvaný "Handshaking".
   (b) Odesílatel vždy čeká odpověď, zda byla data doručena v pořádku.
   (c) Ukončení spojení.

   Tyto body v případě vývoje některých specifických žánrů počítačových her, můžou zpomalovat komunikaci. Následně tato skutečnost může vést k pomalému zobrazování některých důležitých herních částí nebo posunování hráčů (lagování) po herní ploše. V našem případě, není potřeba přenosu dat v reálném čase a zpoždění 100-300 milisekund není problémem.

3.3 Zabezpečení serveru a protokolu

V této části bakalářské práce se zmíním o použitých prvcích, které zabezpečují hru od případného útoku hackerů nebo jen udržují zadané údaje hráčů v bezpečí.

1. Server, na kterém aplikace běží používá heslo s dostatečnou složitostí na to, aby se například přes SSH nikdo nepřipojil a neměl tak přístup ke všem zdrojovým kódům.

2. Přístup k databázi je řešen tak, aby nebyla možná manipulace s daty z třetí strany, SQL injection a podobně.

3. Server podporuje bezpečnostní prvky SSL certifikátu:

   Po připojení klienta na server, je následná komunikace šifrovaná a je tedy zabezpečená od případných útoků. Navíc SSL používá asymetrickou šifru což znamená, že obě komunikující strany mají k dispozici dva klíče a to veřejný a soukromý. Pomocí veřejného klíče můžeme zašifrovat danou zprávu a pomocí soukromého klíče, který nikdo jiný nezná ji dešifrovat.

   K tomu abychom zabezpečili komunikaci pomocí SSL certifikátu, bylo nutné nejdřív takovýto certifikát vygenerovat. Na internetu existuje spousta generátorů, kterému zadáte jakési údaje, většinou se jedná o jednoduché otázky na které odpovíte a on podle nich vygeneruje soubor, který obsahuje požadované šifry. Tento soubor je speciálně upraven, aby nebyl čitelný (je zahashován) a je nutné jej přiřadit jak serverové části aplikace, tak klientské části.

4. Upravení odesílaných dat tak, aby byla komunikace zabezpečena:

   Jelikož se po síti posílají citlivá data, a my nechceme aby byla získána někým cizím, vymyslel jsem jednoduchý postup. Data transformované tak, aby nebyla pouhým okem čitelná a navíc, byla zakódována tak, aby nebyla zpětně rozeznatelná bez použití předchozího postupu, který byl proveden před tím než byla odeslána.

   Pro převod jsem použil tak zvaný base64 formát, kde se jedná o převod binárních dat do ASCII znaků.

   Posíláme takovýto řetězec:

   ```
   {”type”:”request”,”method”:”/getRooms”,
    ”params”:[],”arrayParams”:[]}
   ```

   po převodu na base64 dostaneme takovýto řetězec:

   ```
   VzcG9uc2UiLCJtZXRob2QiOiJzaG93Um9vbXMiLCJwYXJhbXMiOltdLCJhcn
   JheVBhcmFtcyI6W1s1XSxbIjF2MSJdXX0=eyJ0eXBlIj
   ```

   Tomuto převedenému řetězci se říká heš(hash).

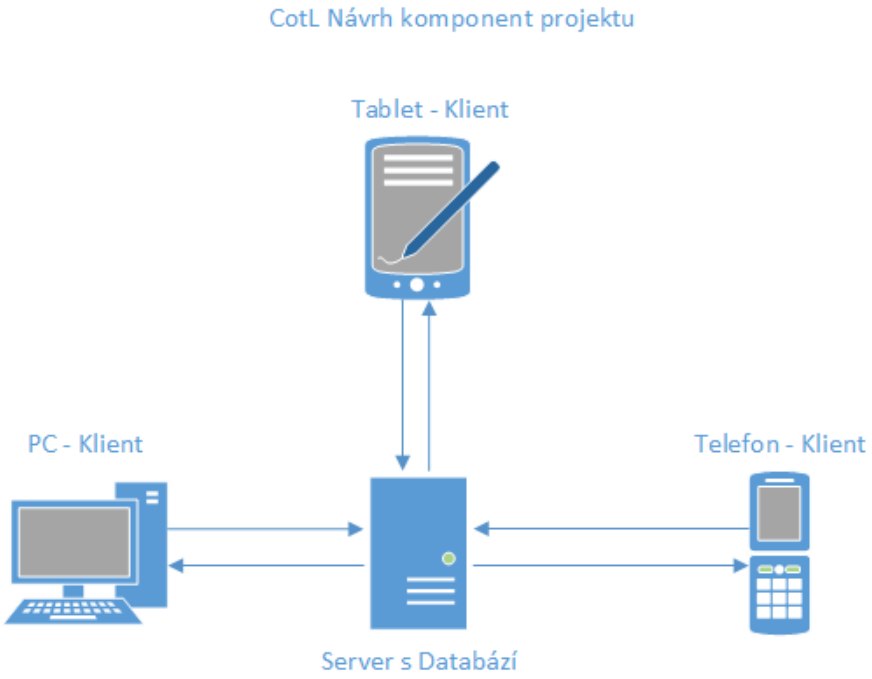
   Jelikož v případě útoku na server by jednoduše útočník mohl získat zpět data zpětným převedením na String, provedl jsem tedy nad touto změtí symbolů některé úpravy, které by útočník neměl zjistit. Tím zůstávají data chráněna.

4 Návrh a implementace serveru

V této kapitole zmíním postup vývoje serverové části, instalaci virtuálního serveru a potřebných komponent a navržený protokol použitý při samotné implementaci serverové části projektu. Zabezpečení serverové části a protokolu proti útokům a některá herní opatření, která serverová část řeší, jako je na příklad odpojení hráče z rozehrané hry nebo nečekaný výpadek sítě.

4.1 Návrh serveru

Před samotným začátkem implementace jsem se rozhodl nakreslit si schéma komponent 2, které budou součástí projektu.



Obrázek 2: Klient - Server Architektura - schéma komponent projektu

Ze schématu bylo patrné, že je potřeba mít server, na kterém bude permanentně spuštěna serverová část projekt, a databázový server, prostor pro tvorbu databáze. Zakoupil jsem tedy virtuální server od společnosti Wedos, na kterém provozuji jak serverovou část tak nainstalovaný mySQL server, který slouží jako databázový server.

4.1.1 Instalace virtuálního serveru

Po zakoupení virtuálního serveru, jsem zvolil automatickou instalaci Debian Linuxu, která sice vybraný operační systém nainstalovala, ovšem bylo třeba mnoho úprav, aby

byl server připraven jak pro spuštění aplikace, tak aby sloužil jako mySQL Server.
Pro přístup na daný server jsem se rozhodl použít volně použitelný program WinSCP, který používá výše zmiňovaný protokol FTP pro vzdálený přístup k souborovému systému. Pro přístup k příkazové řádce samotného serveru bylo potřeba doinstalovat aplikaci pro komunikaci skrz protokol SSH, proto jsem přidal do aplikace WinSCP, aplikaci PuTTy, která zajišťuje SSH komunikaci.
Nyní když jsem byl schopen na dálku ovládat příkazovou řádku serveru, mohla začít instalace důležitých komponent přímo na server.

1. Změna SSH hesla:

   Pomocí příkazu `passwd` jsem změnil heslo pro přístup k serveru, hlavně z důvodu bezpečnosti, jsem zadal heslo tvořené ze směsice náhodných znaků a čísel.

2. Aktualizace systému:

   Pro aktualizaci systému slouží příkazy `apt-get update` a `apt-get dist-upgrade` a po úspěšné aktualizaci bylo potřeba restartovat server příkazem `reboot`.

3. Instalace MySQL Serveru:

   Příkazem `apt-get install mysql-server mysql-client` začala instalace my SQL serveru. Po úspěšném nainstalování jsem zjistil, že aby MySQL server nezahltil výkon celého serveru, bude nutná úprava konfiguračního souboru umístěného v adresáři `/etc/mysql/my.cfg`.

   Zde bylo nutné nastavit komunikační port MySQL, který je v základu 3306, nastavení maximální velikost příchozích a odchozích paketů a tak dále, abychom předešli nechtěným problémům.

4. Instalace Adminera

   Po úspěšné instalaci MySQL serveru, jsem se rozhodl nainstalovat webové rozhraní pro práci s databází, jelikož by bylo nepříjemné vytvářet tabulky a spravovat je přímo v MySQL příkazové řádce, hledal jsem nástroj, který mi pomocí grafického rozhraní zjednoduší práci nad databází. Pro MySQL existuje program jménem Adminer, který je napsaný v jazyku PHP a perfektně splňuje mé požadavky.

4.1.2 Návrh serverové části aplikace

Před implementací serverové části jsem se zamyslel nad tím, jak by měl takový server vlastně fungovat. Určil jsem si tedy požadavky, které na serverovou část kladu.
Specifikace požadavků Serverové části:

1. Server bude moci komunikovat z připojenými klienty, neustále bude naslouchat příchozím požadavkům.

2. Pro každého nově příchozího klienta server vygeneruje nové vlákno.
3. Na server se bude moci připojit omezený počet klientů.
4. Server jako jediný bude komunikovat s databází.
5. Pro komunikaci mezi klientem a serverem, bude použito TCP spojení, a zprávy ve speciálním formátu.
6. Server bude sloužit k registraci a přihlášení do hry více hráčů.
7. Server umožní vytvářet herní místnosti, umožní zobrazit vytvořené herní místnosti, připojit se do konkrétní místnosti a opustit místnost, či hru.
8. Po splnění požadavků umožní server nastartovat hru a předávat potřebné parametry mezi klienty.

Stručný popis jednotlivých bodů, požadavků:

1. Serveru byla nastavena ip adresa a port takový, aby mohla být komunikace zajištěna a server mohl hostit jednotlivé připojené klienty
2. Pro každé nově příchozí spojení je vygenerováno nové vlákno, ve kterém se obsluhuje právě připojený klient. Bylo nutné použití vláken z toho důvodu, že potřebujeme v jeden čas obsluhovat více klientů současně.
3. Serveru je přidělený pevně daný počet maximálně připojených uživatelů, proto aby server nebyl přetížen.
4. Server obsahuje třídu pro připojení ke zmiňované MySQL databázi která mu dovoluje práci nad jejími daty.
5. Komunikace mezi klientem a serverem, je řízena TCP protokolem, a zprávy které se zasílají mají speciální formát a to dvojici parametrů Typ a Metoda. Za nimi následují data, které chceme společně s dotazem poslat. To vše se převede pomocí knihovny Gson na formát typu json. Finálně se tedy jedná o textový řetězec reprezentující danou zprávu.

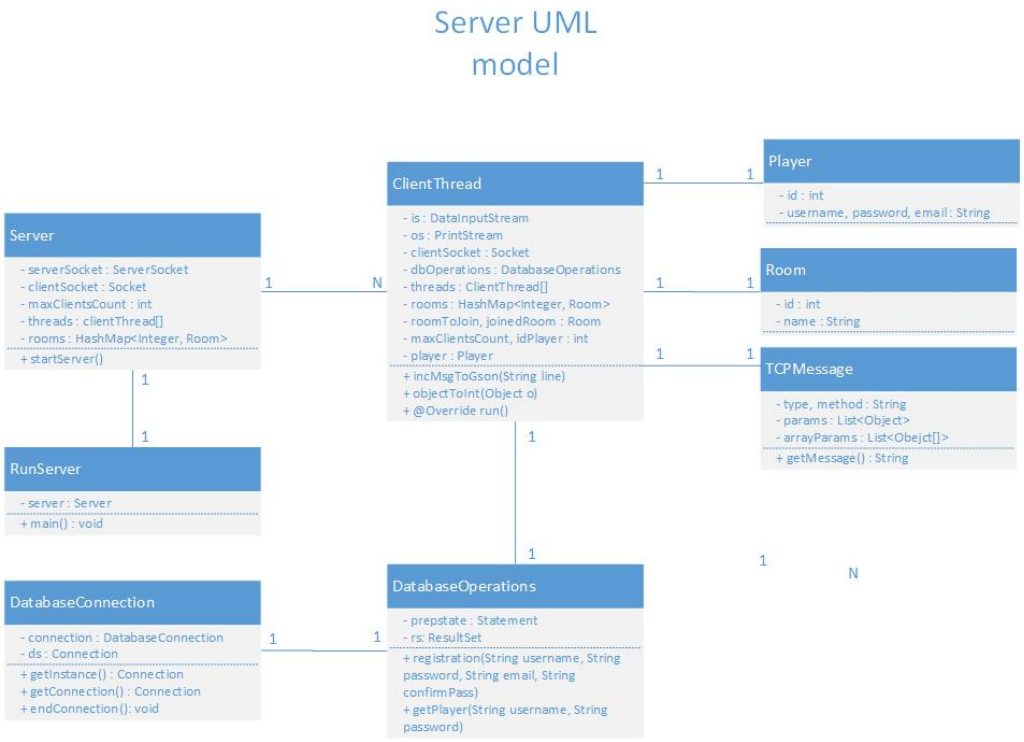
   Příklad 4.1
   Klient žádá o výpis aktuálně vytvořených místností.

   - Klient zasílá žádost na server ve formátu ”request”,”/getRooms”
   - Server jej příjme, a podle parametru ”/getRooms”zjistí co má dělat.
   - V tomto případě zjistí zda existuje alespoň jedna založená místnost, pokud ne odešle na klienta zprávu ”response”,”noRoomsAvalible”+ přidá parametr reprezentující hlášku ”There are no rooms avalible.”.
   - V opačném případě projde všechny místnosti, a odešle je na klienta.

6. Pro hraní hry po internetu je nutná registrace účtu a následné přihlášení do hry. Z důvodu manipulace s hráči a zabránění nechtěného přístupu do hry.
7. Rozhodne li se hráč hrát s ostatními hráči po internetu, nejprve musí vytvořit herní místnost nebo se připojit do existující, a počkat než se naplní. Hráč může opustit místnost, stejně tak i rozehranou hru.
8. Po naplnění místnosti lidmi, se začne odpočítávat 10 vteřin, než začne samotná hra. Server se pak stará o to, aby hráči měli vždy aktuální informace o tom, která karta byla použita a na koho.

4.2 Implementace serverové části

V této části detailně přiblížím průběh vývoje serverové části a podrobně popíšu každou třídu obsaženou v serverové části. Vývoj serverové části nejlépe nastínim na UML diagramu, obrázek č.3



Obrázek 3: UML diagram - serverová část

4.2.1 Třída RunServer

Tato třída slouží pouze k vytvoření instance třídy Server a spuštění aplikace, proto obsahuje metodu main, ve které je pouze zavolána metoda startServer(), která je metodou třídy Server. Tato metoda má na starosti právě spuštění serveru, a více se jí budeme věnovat v podkapitole Třída Server.

4.2.2 Třída RunServer

Jedná se o třídu, ve které se spravuje příchozí spojení, a vytváří se zde pro každého nového klienta nové vlákno.

Použité atributy:

- `ServerSocket serverSocket` - Tomuto atributu přiřadíme port, na kterém chceme aby server poslouchal.
- `Socket clientSocket` - Jedná se o socket příchozího spojení, tudíž klienta, který se snaží připojit na server.
- `MaxClientsCount` - Maximální počet připojených klientů.
- `ClientThread[] threads` - Při každém nově příchozím spojení, se vytvoří vlákno pro daného klienta a to se uloží do pole threads, které pak slouží jako úložiště všech připojených uživatelů.
- `HashMap<Integer,Room> rooms` - Tato hashmapa slouží k evidenci všech vytvořených místností, proto aby všichni hráči měli přistup ke všem místnostem.

Použité metody:

- `void startServer()` - V rámci této metody se nastartuje server a stále se čeká na nově příchozí klienty, pokud je server plný, to znamená že počet připojených klientů je roven maximálnímu počtu připojených klientu, server vypíše hlášku ”Server is too busy. Try later.”.

4.2.3 Třída ClientThread

Třída dědící z třídy Thread, s upravenou metodou run(), která se spustí pro každého nově připojeného klienta.
Použité atributy:

- `DataInputStream is` - Slouží pro přečtení příchozích zpráv zaslaných od klienta.
- `PrintStream os` - Slouží k zasílání zpráv na klienta.

- `DatabaseOperations dbOperations` - Instance třídy, která slouží pro volání metod na databázi, na příklad registrace nebo přihlášení.
- `Room joinedRoom` - Zde si udržuji, ve které místnosti je daný uživatel připojený.
- `Player player` - Instance třídy Player, do které po přihlášeni uložím údaje o hráči z databáze.

Použité metody:

- `incMsgToGson(String line)` - Metoda sloužící pro převod přijaté zprávy, která je ve formátu Json, což reprezentuje string, na objekt typu TCPMessage, pomocí metod z použité knihovny Gson. Objekt typu TCPMessage pak obsahuje typ zprávy, zda-li se jedná o žádost nebo odpověď, druhý atribut metodu, co s příchozími daty chceme dělat, a případně další atributy nebo pole atributů, se kterými se pak pracuje.

Definice metody:

```
public TCPMessage incMsgToGson(String line){
        Gson gson = new Gson();
        TCPMessage tcpmess = gson.fromJson(line, TCPMessage.class);
        return tcpmess;
    }
```

Výpis 1: Příchozí zpráva na objekt.

@Override
public void run(): tato metoda obsahuje nejdůležitější část komunikace, jelikož jakmile se připojí uživatel, spustí se právě tato metoda. Obsahuje nekonečnou smyčku, která stále poslouchá, zda klient neposlal nějakou zprávu. Stane-li se tak, zjistí o jaký dotaz se jedná a podle toho jedná.
Ukázka naslouchající části, kdy server řeší přihlašování do hry.

```
if (incMsgToGson(line).getType().equals("request")){
    TCPMessage msg;
     switch(incMsgToGson(line).getMethod()){
       case "/login":
           String lResult = null;
              msg = new TCPMessage("response","login");
              Player player = dbOperations.getPlayer(
                  (String)incMsgToGson(line).getParams(0),
                  (String)incMsgToGson(line).getParams(1));
              if (player.getId() != 0){
                 this.player = player;
                  lResult = "Sucessfully loged in as player " + player.getId();
                  setIdPlayer(this.player.getId());
              }
              else {lResult = "Please check your username or password and try again.";}

              msg.addParam(lResult);
              msg.addParam(player.getId());
```

```
                os. println (msg.getMessage());
            break;
        }
    }
```

Výpis 2: Příchozí žádost o přihlášení.

Pokud se jedná o zprávu typu "`request`" a zároveň o metodu `/login`(přihlášení), server zavolá metodu `dbOperations.getPlayer()` která vrátí instanci hráče. Pokud hráč v databázi existuje a zadal správné přihlašovací údaje, server odešle zprávu zpět na klienta, jestli bylo přihlášení úspěšné nebo ne.

4.2.4 Třída TCPMessage

Třída která byla vytvořena za účelem reprezentovat formát zaslaného paketu. K tomu abychom objekty této třídy převedly na řetězec znaků, který následně můžeme zaslat na požadované zařízení. K tomu jsem se rozhodl použít knihovnu od společnosti Google, která ji pojmenovala Gson. Tato knihovna má na starosti dvě věci, serializaci objektu TCPMessage do formátu Json, což je zmiňovaný String. Druhá věc kterou má na starost, je deserializaci zpět na objekt typu TCPMessage. K serializaci slouží metoda `getMessage()`, a k deserializaci slouží metoda `incMsgToGson()`.
Použité atributy:

- `String type` - reprezentuje typ, zda se jedná o žádost nebo odpověď "request"- žádost "response"- odpověď.
- `String method` - jedná se o upřesnění, o co se vlastně jedná, na příklad o žádost o přihlášení nebo odpověď.
- `List<Object> params` - slouží k přiřazení parametrů, které chceme odeslat společně s žádostí. V případě přihlášení se jedná o uživatelské jméno a heslo.
- `List<Object[]>arrayParams` - slouží k přiřazení pole parametrů, které chceme odeslat ve zprávě. Na příklad místnosti které jsou k dispozici.

Použité metody:

- `public String getMessage()` - Slouží k sestavení zprávy. Tato metoda převede objekt TCPMessage, který je složen z výše uvedených atributů, na tak zvaný Json, což reprezentuje komunikační zprávu ve formě String řetězce.

Definice metody:

```
public String getMessage(){
    Gson gson = new Gson();
     String json = gson.toJson(this);
     return json;
}
```

Výpis 3: TCP Message na Json.

4.2.5 Třída Room

Jedná se o třídu reprezentující herní místnost ve hře. Do této místnosti se pak napojí daný počet lidí a odstartuje se hra.
Použité atributy:

- `int id` - Jedná se o náhodně vygenerované číslo, z důvodu bezpečnosti. Kdyby chtěl někdo napadnou místnosti podle ID, měl by jednoduché tak učinit, kdyby se číslovaly od jedné a dál.
- `String name` - Jméno místnosti, které si volí hráč při založení místnosti.

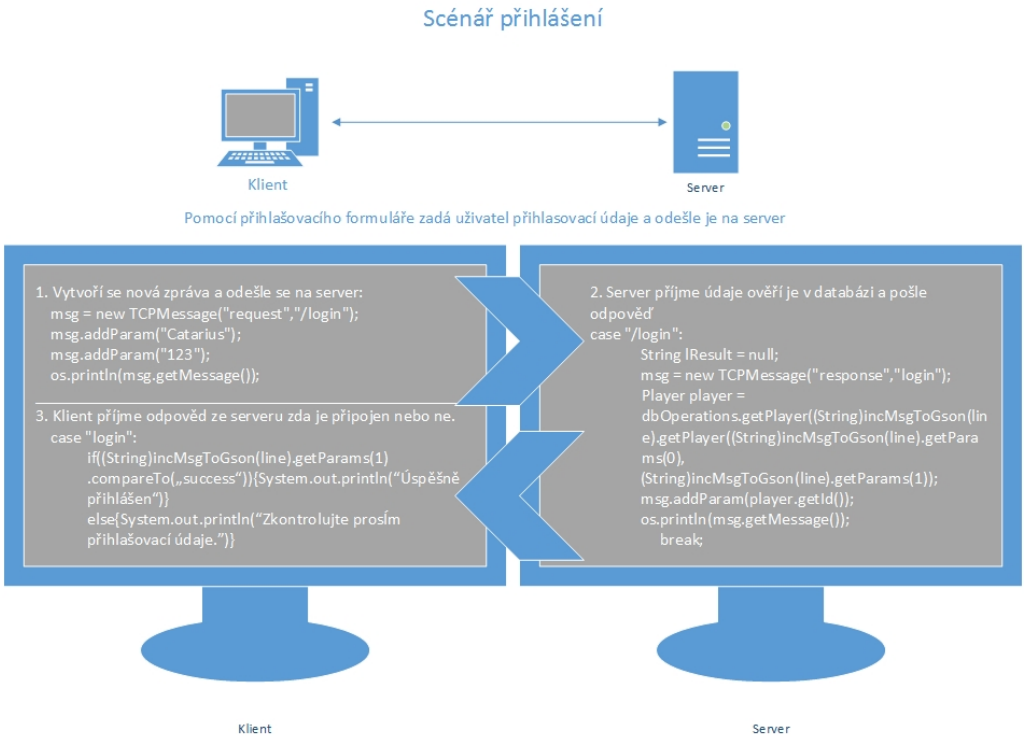
4.2.6 Třída Player

Třída reprezentující záznam hráče z databáze. Jedná se o ORM k tabulce Player a pomocí id následně identifikujeme o koho se jedná, po případě na koho zasíláme kartu.
Použité atributy:

- `int id` - Id hráče z databáze.
- `String username, password, email` - Jedná se o registrační údaje, které je nutné vyplnit při registraci. Při přihlašování stačí pouze username a password.

4.3 Návrh komunikačního protokolu

Jedním z úkolů bakalářské práce je navrhnout takový protokol, který je vhodný pro client-server komunikaci. Po delším zkoumání, které jsem do částečné míry osvětlil v části 3.2 Komunikační protokoly, jsem se rozhodl pro použití TCP protokolu. TCP protokol splňuje všechny předpoklady a pokud upravíme odesílané zprávy do srozumitelného formátu, může být vytvořená herní API čitelná i ostatními programátory. Jelikož skrze sokety můžeme zasílat data v textovém formátu String, vymyslel jsem metodu, která je jednoduchá a zároveň efektivní jak pro programátory, tak pro procesor.
Třída TCPMessage:
Tuto třídu podrobně popisuji v podkapitole 4.2.4 Třída TCPMessage odvolávám se tedy k této podkapitole a v krátkosti zrekapituluji co má vlastně na starost. Třída TCPMessage reprezentuje jednu zprávu, která bude zaslána. Obsahuje dva povinné parametry ”type a method”, kde typem se myslí zda se jedná o žádost nebo odpověď metodou o co přesně se jedná. Příkladem může být ”/login”čímž se myslí přihlášení. Následují nepovinné parametry, kterými jsou právě data, které chceme odeslat.
Takto vygenerovaný objekt, kterému byly určeny všechny parametry následně převedeme na řetězec znaků, tomuto procesu se říká serializace. Výsledný Json řetězec odešleme na požadované zařízení. Toto zařízení příjme Json řetězec a převede je zpět na objekt typu TCPMessage, čímž máme přístup ke všem zaslaným atributům. Tento proces má na starosti metoda `incMsgToGson()`.



Obrázek 4: Princip komunikace

1. Klient vyplní přihlašovací formulář, zde zadal údaje, přihlašovací jméno ”admin”, heslo ”nereknu”a stiskl tlačítko přihlásit tudíž se vygenerovala zpráva žádost, o přihlášení(1), předání údajů(2) a zkompletování a následné zaslání na server(3).

   (1) `msg = new TCPMessage(”request”,”/login”);`

   (2) `msg.addParam(”admin”);`

   (2) `msg.addParam(”neřeknu”);`

   (3) `os.println(msg.getMessage());`

2. Server příjme požadavek a předvytvoří zpravu pro odpověď(4), zjistí z databáze, zda exis-tuje uživatel se zadaným jménem a heslem(5), přidá ke zprávě hráčovo identifikační číslo(6) a odešle zprávu na klienta(7).

   (4) `msg = new TCPMessage(”response”,”login”);`

   (5)
```
Player player = dbOperations.getPlayer(
(String)incMsgToGson(line).getPlayer(
(String)incMsgToGson(line).getPara ms(0),
(String)incMsgToGson(line).getParams(1));
```

   (6) `msg.addParam(player.getId());`

   (7) `os.println(msg.getMessage());`

3. Klient příjme odpověď ze serveru a pokud je přijaté id hráče větší než 0, což znamená že existuje přihlásí uživatele(8), jinak vypíše hlášku "Zkontrolujte prosím přihlašovací úda-je."(9).

   (8) `if(objectToInt(incMsgToGson(line).getParams(1))>0) System.out.println(\Úspěšně přihlášen\)`

   (9) `else {System.out.println(\Zkontrolujte prosím přihlašovací údaje.")}`

Komunikace, která mezi účastníky probíhá je šifrovaná asymetrickou šifrou pomocí SSL certifikátu, tudíž je chráněná vůči vnějším útokům.
Odeslané zprávy jsou taktéž převedeny do base64 formátu a výsledný hash transformován tak, aby útočník nevěděl jak z něj zpětně dostat data.
Systém je tudíž dostatečně chráněn proti nepříznivým jevům a osobám.

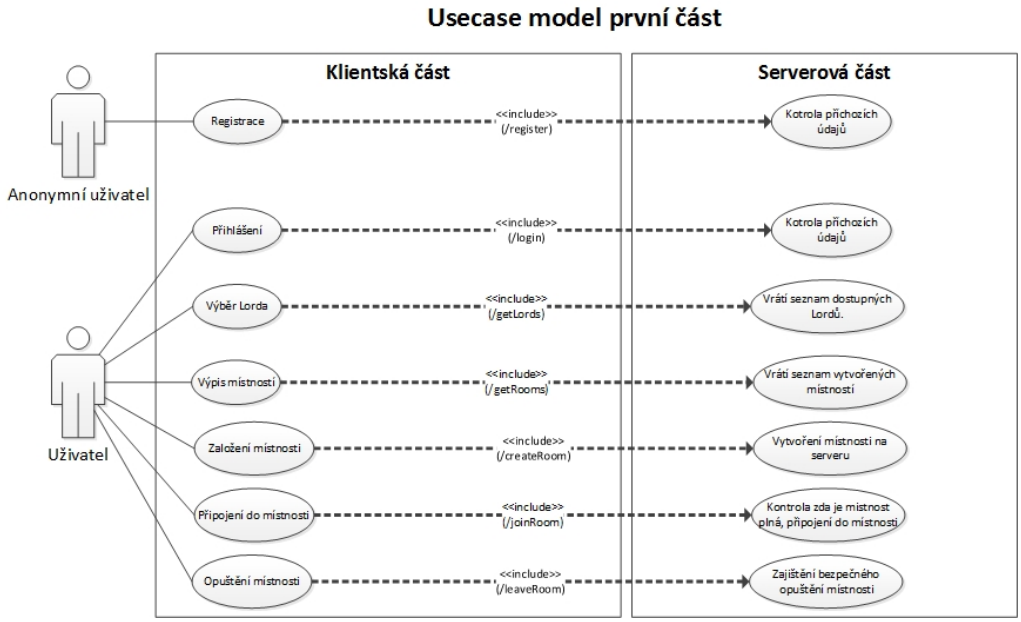
5 Návrh a implementace hry na PC

Poslední část bakalářské práce se zabývá implementací samotného herního klienta na PC. Pro vývoj jsem zvolil zmiňovaný programovací jazyk c# a vývojové prostředí Visual Studio 2012. Herní klient obsahuje menu, přihlašovací formulář, registrační formulář a samotný herní formulář, ve kterém je spuštěna hra. Jelikož jsem se návrhem hry zabýval již v sekci 2.1 Návrh hry, odvolávám se k této kapitole a v rámci této poslední části ukážu některé případy užití a uživatelské rozhraní UI.

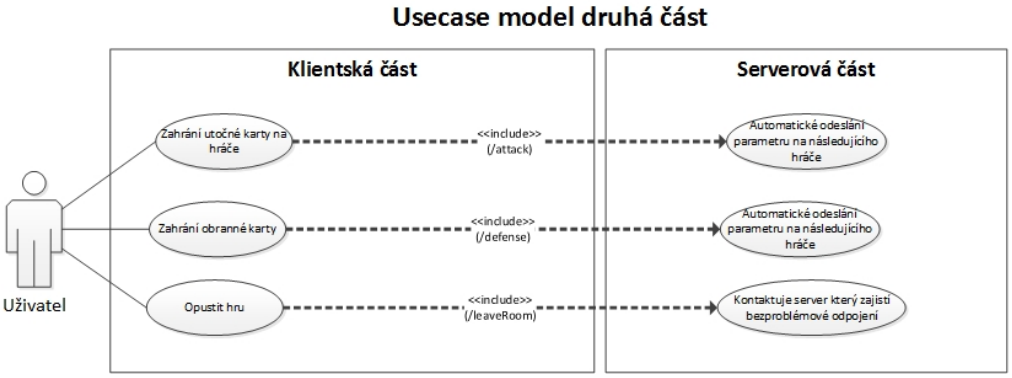
5.1 Případy užití

Zde si ukážeme některé části hry, které považuji za důležité, k nim přidám sekvenční diagramy pro názorné představení situace. Nechybí zde ani Use Case model celého herního klienta.
Herní klient jako takový obsahuje mimo zdrojového kódu, který zajišťuje průběh hry také různé příkazy. Ty se po jejich zavolání spustí a zašlou na server, ten pak pracuje s odeslanými daty a chová se přesně podle toho, co klient chce. Proto zde existuje nespočet metod, kde každá z nich má za úkol něco jiného. Zde je ukázka některých zajímavých příkazů.

5.1.1 Use case model herního klienta a serveru



Obrázek 5: Usecase model první část



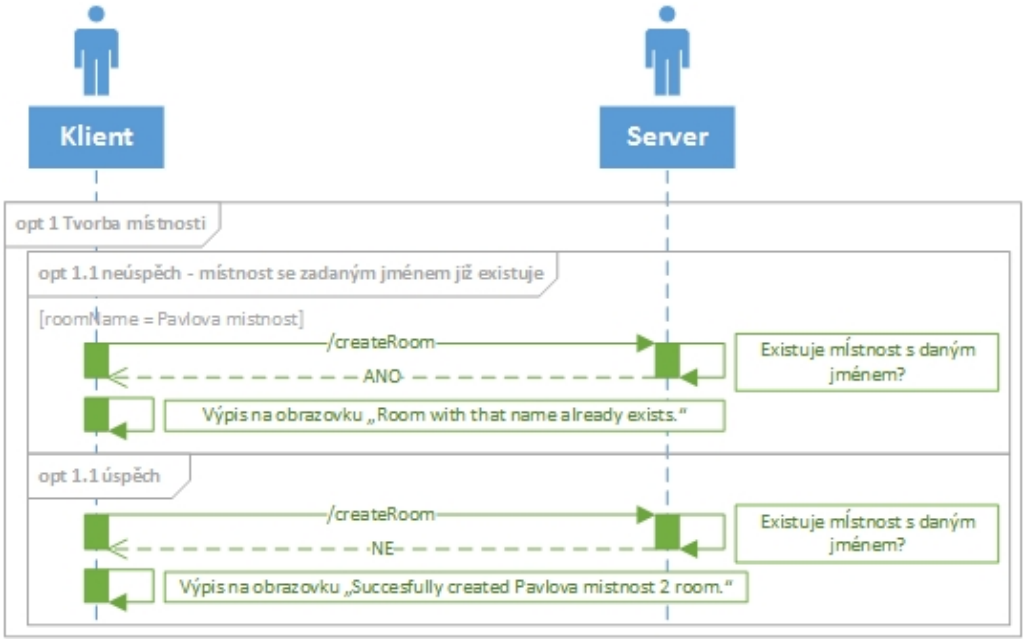
Obrázek 6: Usecase model druhá část

5.1.2 Správa místností

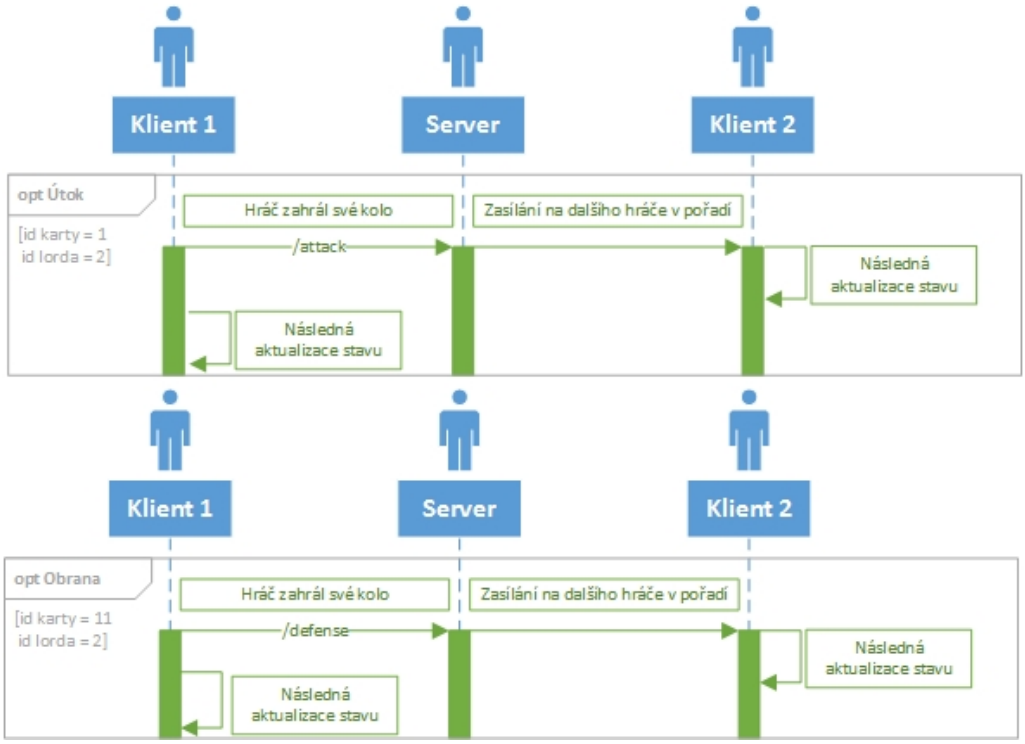
Po přihlášení se zobrazí seznam aktuálně vytvořených místností. Tento seznam se dá aktualizovat pomocí tlačítka Refresh. Pomocí tlačítka Create Room máme možnost vytvořit mistnost, do které můžou vstoupit až 4 hráči. Jakmile se tak stane, začne se odpočítávat 10 sekund a pokud nikdo z přítomných neodejde z místnosti pomocí tlačítka Leave Room, hra se sama zapne. Tímto jsme se dostali k tlačítku Leave Room. Toto tlačítko bezpečně vyjme hráče z místnosti a vrátí jej do nabídky výběru místností. A tlačítko Join Room slouží pro připojení do vybrané místnosti, pokud je místnost, plná tak se zobrazí hláška že se již nemůžeme připojit a musíme tudíž zvolit jinou místnost. Po připojení do místnosti se zobrazí seznam hráčů, kteří v místnosti jsou a se kterými budeme hrát.
Při vytváření místnosti se objeví dialog, kde zadáme jméno místnosti a pokud místnost s daným jménem ještě neexistuje, vytvoří se na serveru daná místnost a vygeneruje se náhodné číslo reprezentující id místnosti. Náhodné číslo z důvodu bezpečnosti, jako ochrana proti hackerům.

5.1.3 Průběh hry

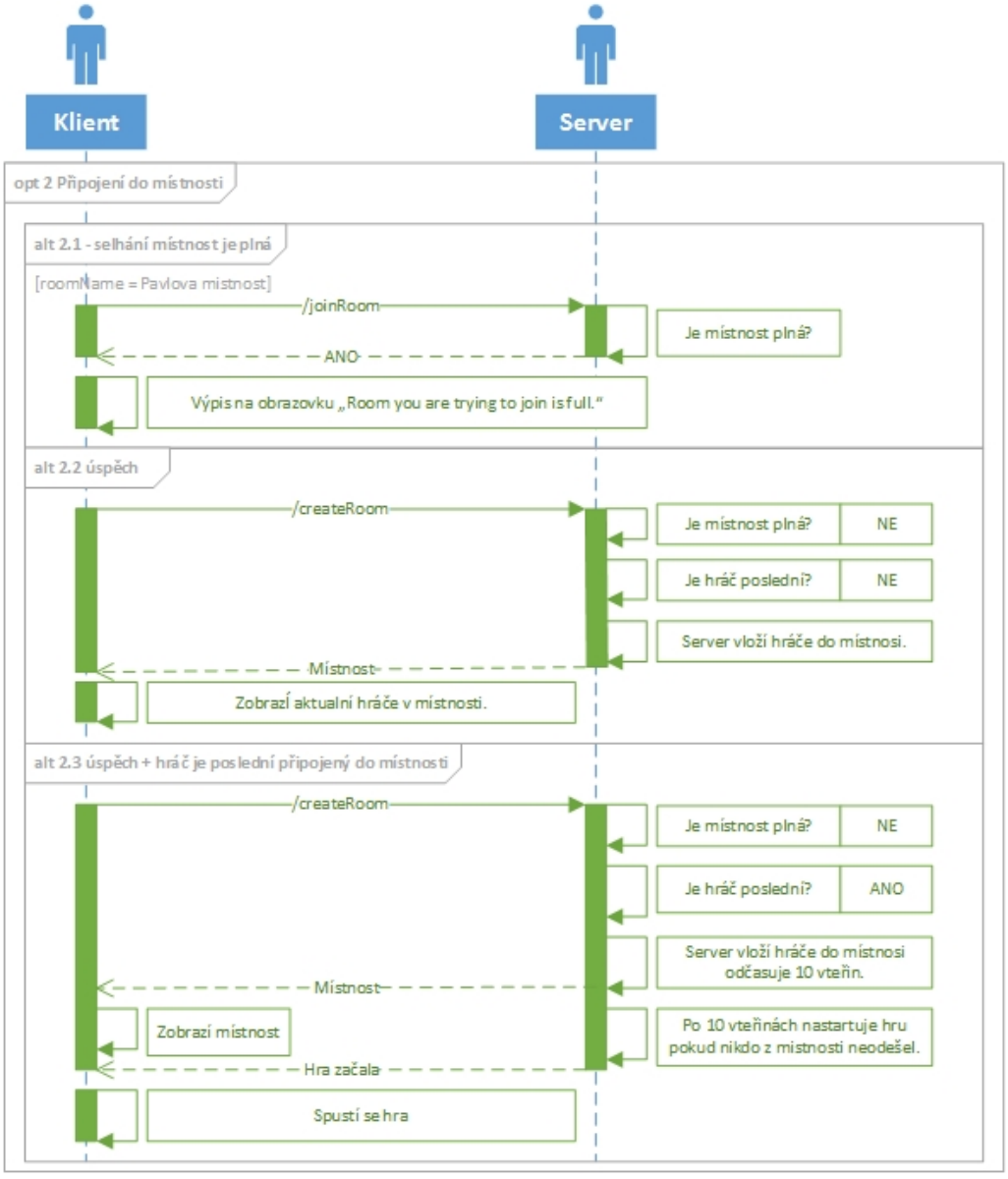
Hra probíhá tak, že na obrazovce je vykreslen aktuální stav všech hráčů, avšak daný hráč útočí vždy na hráče bezprostředně následnujícího za nim. To znamená, že hrají-li 4 hráči, první hráč útočí na druhého, druhý na třetího, třetí na čtvrtého a čtvrtý na prvního. V případě vyřazení jednoho z hráčů, se změní pořadí a tudíž i protivník na kterého útočíme. Po odstartování kola má hráč patnáct sekund na to, aby zahrál své kolo jinak bude vyhozena náhodná karta z jeho ruky. Hráč má dále možnost hrát defensivně nebo útočně a to pouze jednou kartou.
Proces komunikace mezi klienty na sekvenčním diagramu obrázek č. 8



Obrázek 7: Sekvenční diagram - Tvorba místnosti



Obrázek 8: Sekvenční diagram - Útok a obrana



Obrázek 9: Sekvenční diagram - Připojení do místnosti

6 Grafické rozhraní

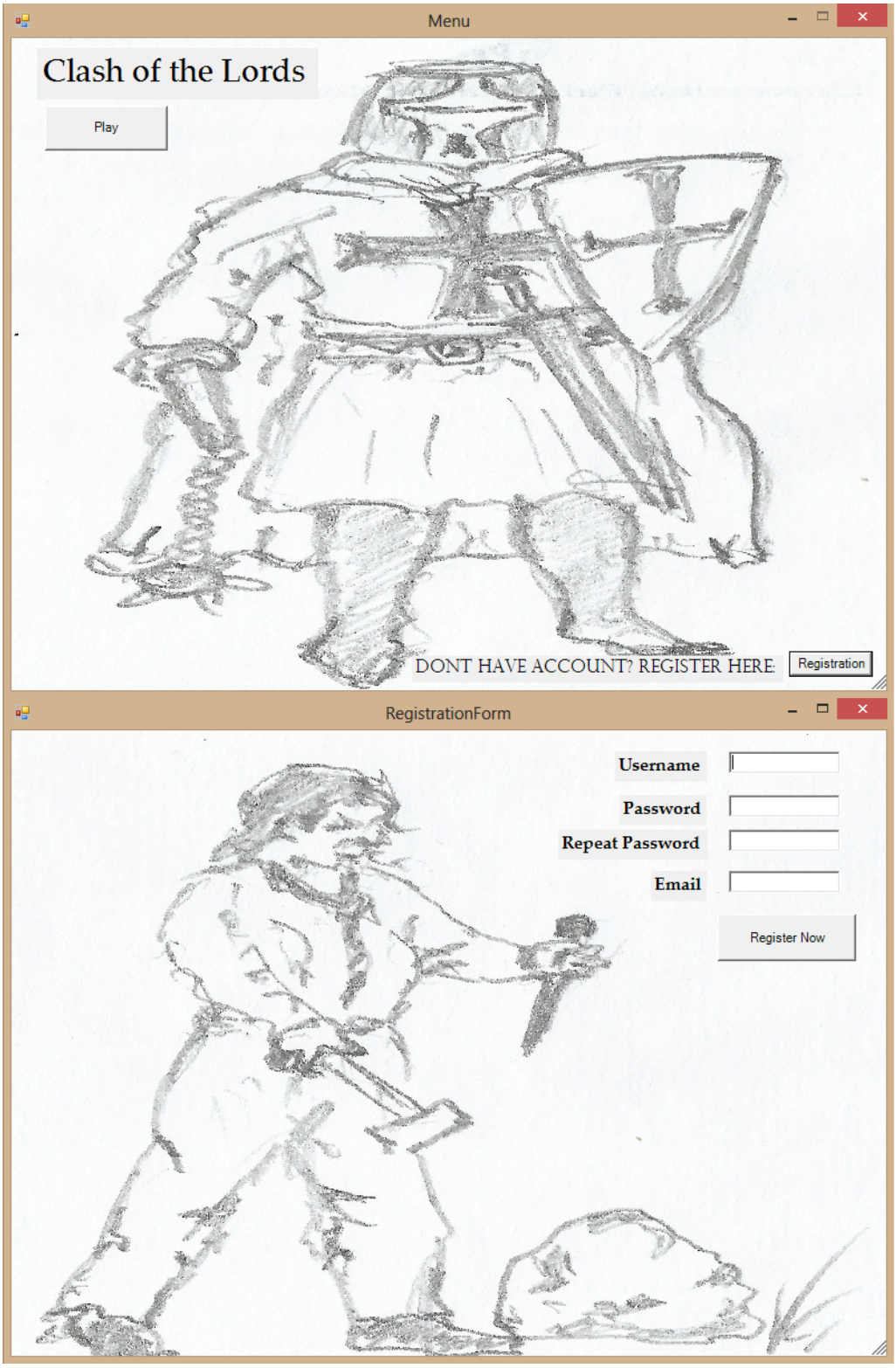
V této sekci se nachází popis uživatelského rozhraní, jak před začátkem hry (Menu) tak v průběhu hry. Ukázka uživatelského rozhraní v podobě obrázků a stručný popis akcí, které se stanou po stisku jednotlivých tlačítek.

6.1 Menu UI

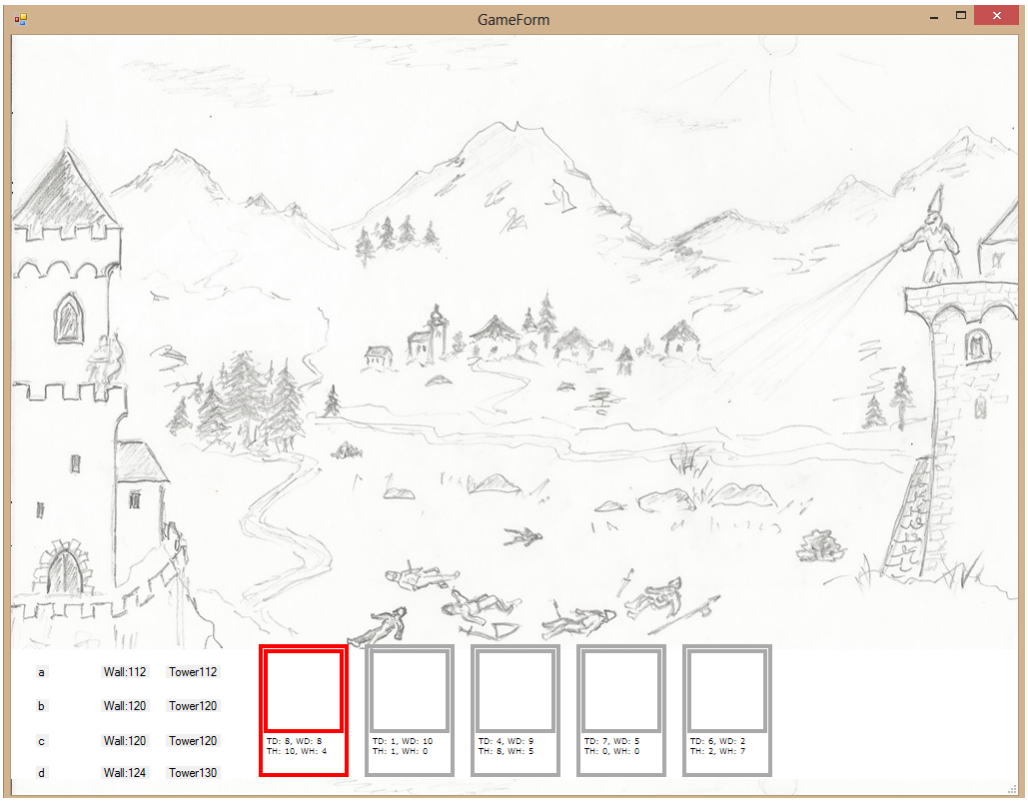
Po zapnutí hry se objeví hlavní menu viz obrázek 10. Toto menu obsahuje tlačítka Registration a Play.

1. Menu formulář:
   - Registration - Otevře formulář registrace. Formulář po zadání údajů a stisku tlačítka Register Now odešle na server příkaz registrace. Server zpracuje přijatá data a pokud se shodují zadaná hesla, odešle příkaz na databázi, uloží uživatele permanentně do databáze a vrátí na klienta výsledek, zda byl úspěšně zaregistrován nebo ne.
   - Play - Spustí formulář pro přihlášení do hry - Login Form. Po přihlášení se objeví formulář Rooms s aktuálně založenými místnostmi. Zde se nachází tři tlačítka Create Room, Join Room a Refresh.
2. Login formulář
   - Submit - Potvrzení přihlášení Server příjme údaje a dotáže se databáze zda daný uživatel existuje. Pokud ano, odešle zprávu že byl uživatel úspěšně přihlášen, pokud ne odešle zprávu,kde žádá o překontrolování údajů.
3. Room formulář:
   - Create Room - Zobrazí dialog pro zadání jména místnosti, a tlačítko Create Room, které definitivně odešle zprávu na server, která vytvoří místnost na serveru.
   - Join Room - Vybereme-li si místnost ze seznamu a stiskneme tlačítko Join Room, odešle se zpráva na server a server zajistí, že se připojíme do vybrané místnosti. Navíc se čeká dokud místnost nebude plná, poté začne hra.
   - Refresh - Načte aktuální místnosti ze serveru a zobrazí je do tabulky.
4. Herní Formulář:

   Po připojení všech hráčů do místnosti se po deseti vteřinách odstartuje hra a všem hráčům se vygeneruje herní pole. Toto pole obsahuje náhodně vygenerované karty, které může hráč používat libovolně, po kliknutí na ně. V levo se udržuje aktuální skóré a na obrázku č. 11 si můžeme povšimnout, že hrají hráči ”a,b,c,d”. Jedná se o testovací hráče a jejich skóre se mění v závislosti na použité kartě.



Obrázek 10: UI - Menu, Registrační formulář



Obrázek 11: UI - Herní formulář

7 Zhodnocení použitého postupu vývoje

Aplikace jsem vyvíjel průběžně, jelikož jsou závislé jedna na druhé. Nejprve jsem však místo hry vyvíjel konzolovou Java aplikaci, kterou jsem testoval jednotlivé příkazy na server. Zprvu jsem zhotovil funkční TCP spojení tak, aby server přijímal nová spojení a hostil je ve speciálních vláknech. Toto vlákno, které bylo vygenerováno speciálně pro každého nově příchozího klienta obsahuje nekonečnou smyčku, která čeká na žádosti od klien-tů. Klient taktéž obsahuje nekonečnou smyčku, kterou odposlouchává odpovědi ze serveru.
Takto vytvořené aplikace jsem obohacoval stále o nové schopnosti, na příklad server jsem připojil k databázi a mohl jsem řešit práci s databází. Postupně jsem přidával nové funkce jak na stranu klienta, tak na stranu ser-veru.

7.1 Problémy při implementaci

Po úspěšném dokončení všech důležitých funkcí, které musela daná hra splnit, jsem přešel na předělání této java aplikace na c# . Zde jsem musel vyřešit pár problémy, které nastaly při přepisování kódů:

- Navázání spojení: V Javě se připojíme na server pomocí Socket třídy v c# pomocí třídy TcpC-lient. Obě metody jsou téměř podobné.
- Serializace / Deserializace TCP zprávy: V Java aplikaci jsem používal importovanou knihovnu Gson pro serializaci a deserializaci. Tuto knihovnu však v c# nejde použít, protože je psaná v Javě a proto jsem musel zjistit jak se pracuje s serializací do Json formátu a jak jej zpátky deserializovat.
- Zasílání dat: V obou jazycích existují input a output streamy, které slouží pro příjem a odesílání zpráv. V jazyce c# je však nutnost po vložení do output streamu ještě popostrčit data pomocí metody flush(). Navíc pojmenování tříd zajišťujících komunikaci je opět odlišné.

Java

- Printstream - což je output stream (odchozí data)
- DataInputStream - což je input stream (příchozí data)

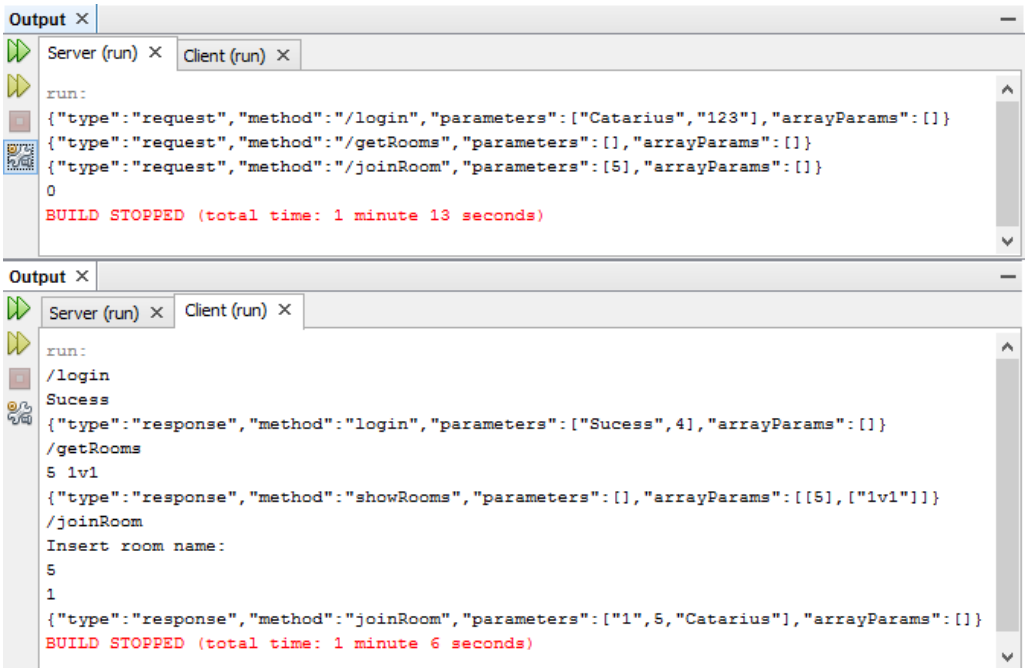
C#

- StreamReader - což je input stream (příchozí data)
- StreamWriter - což je output stream (odchozí data)

Závěrem tedy byla aplikace testována jak pomocí Java konzolové aplikace, tak předělanou c# windows form aplikací. Hra je navržena tak, aby na ní šly otestovat nejdůležitější případy. Nejedná se tedy o finální verzi, avšak o testovací verzi, která slouží k vyobrazení všech nutných naimplementovaných částí.

7.2 Testování programů

Nejlepším způsobem, jak otestovat zda fungují zadané příkazy, bylo právě pomocí konzolové Java klient aplikace.
Příklad testování aplikace:



Obrázek 12: Testování - klient / server strany

Zde lze vidět, pokus o přihlášení `/login` kde na serveru výpis ”Success”značí úspěšné přihlášení. Nebo výpis místností `/getRooms` na serveru výpis ”5”značí id místnosti a ”1v1”název místnosti. A poslední zkoušený příkaz `/joinRoom`, který na serveru úspěšně připojil hráče do dané místnosti.

7.3 Zhodnocení použitých prvků při vývoji

Architektura klient - server je podle mého názoru důležitou částí, kterou by měla každá hra obsahovat. Jedná se o sofistikované řešení problému komunikace mezi klientem a server, či klient s klienty.
Co se týče použitého protokolu, určitě by bylo lepší v projektu použít UDP protokol obohacený o prvky TCP protokolu. I když je UDP protokol mnohem rychlejší a můžeme si jej přizpůsobit pro svoje potřeby, TCP protokol by v tomto případě neměl mít problém hostit velké množství hráčů.
Co se týče připojení na databázi myslím si, že existuje efektivnější způsob jak komunikovat z databází, než že každý klient má své vlastní připojení k databázi. Při více hráčích by mohl tento návrh připojení dělat problémy.

Práce finálně obsahuje plnohodnotnou serverovou část psanou v jazyku Java, uzpůsobenou pro nasazení na server. Herní aplikaci komunikující se serverem napsanou v jazyku c# a klientskou konzolovou část psanou v jazyku Java. Tato hra byla vytvořena pouze pro testovací účely. Pro úspěšné otestování aplikace je nutné spustit serverovou část, poté se připojit se čtyřmi hráči do jedné místnosti. Po deseti vteřinách odstartuje hra kde můžeme otestovat měnící se skóre v závislosti na použitých kartách.

8 Závěr

Shrnutí bakalářské práce:

1. Idea hry Magic (CotL): V této kapitole jsem se zabýval zjišťováním informací, proč vyvíjet zrovna na operační systém Android, jaký žánr hry se v dnešní době ujme, zda by měla hra obsahovat "Hru více hráču"a hlavně proč. Nastínil jsem zde také, které programovací jazyky a nástroje jsou vhodné pro vývoj hry.

2. Technologie: V této části bakalářské práce se vybírala správná architektura. Je zde popsáno proč jsem zvolil právě klient - server architekturu. Na příkladech jsem osvětlil, proč je klient server architektura vhodná pro řešení hry více hráčů a které zařízení mají jaké povinnosti v rámci této architektury. Dále jsem popsal komunikační protokoly, jejich výhody či nevýhody a proč jsem si zvolil právě TCP protokol. Také jsem zde zmínil použité prvky pro zabezpečení serveru, komunikačního protokolu a dat odesílaných z jednoho zařízení na druhé.

3. Návrh a implementace serveru: Okrajově jsem se zde zmínil o tom, jak probíhala instalace virtuálního serveru, aby byl chráněn od útoků a zároveň hostil MySQL databázový server. Podrobně jsem popsal postup implementace serverové části hry a návrh komunikačního protokolu, kde jsem na příkladech ukázal princip SSL komunikace.

4. Návrh a implementace hry na PC: Tato sekce obsahuje základní poznatky při vývoji herního klienta. Dále obsahuje Use Case diagramy popisující důležité části systému, stejně tak sekvenční diagramy popisující nejdůležitější prvky herního klienta.

5. Grafické rozhraní: Zde jsem popsal uživatelské rozhraní jak před začátkem hry(Menu) tak v průběhu hry. Dále jsou zde ukázky klientské části aplikace a podrobný popis tlačítek a příkazů.

6. Zhodnocení použitého postupu vývoje: Na konec, v této kapitole jsem zhodnotil mé snažení, nastínil některé problémy které nastaly při implementaci a ukázal postup testování aplikací.

V bakalářské práci jsem se střetl s mnoha problémy, které jsem musel vyřešit. S většinou problému jsem se setkal poprvé, a proto jejich řešení ne vždy bylo rychlé nebo ideální. Do budoucna by mohla být aplikace doladěna do takového stádia aby se jednalo o bezchybnou hru, obohacenou o více prvků.
Z důvodu nedokončení bakalářské práce Filipa Krupíka hra neobsahuje umělou inteligenci, která měla být převzata. Tím pádem je hra spustitelná pouze proti realným lidem. Přesto všechny body zadání mé bakalářské práce byly splněny.

9 Přílohy

(I.) DVD disk:
SSLServer aplikace(Java)
SSLClient aplikace(Java)
Cotl.Desktop aplikace(C#)
obrázky

10 Reference

[1] Pužmanová, Rita, *TCP/IP v kostce,* České Budějovice, Kopp, 2009.

[2] C# Programming, *C# Programming - Wikibooks, open books for an open world*. http://upload.wikimedia.org/wikipedia/commons/b/b3/C_Sharp_Programming.pdf

[3] Tomas Vilda, *Java SSL komunikace*. http://stilius.net/java/java_ssl.php

[4] Java, MySQL server, *Instalace Java komponent a MySQL serveru*. https://jamfnation.jamfsoftware.com/article.html?id=28

[5] Client - Server Architecture, *Java Client - Server Architecture*. http://www.ase.md/~aursu/ClientServerThreads.html

[6] Alex Berson, *CLIENT/SERVER ARCHITECTURE. 2nd edition*, McGraw-Hill, 1996.

[7] Java, *JSON serialization and deserialization* http://www.mkyong.com/java/how-do-convert-java-object-to-from-json-format-gson-api/

[8] MSDN, *JSON serialization and deserialization* http://msdn.microsoft.com/en-us/library/bb410770.aspx

[9] IDC *Worldwide Quarterly Mobile Phone Tracker*, March 29, 2011

[10] Communication protocols http://en.wikipedia.org/wiki/Communications_protocol

[11] Jan Kroupa, *Obrázky použité v aplikacích*